```
Option Explicit

Dim Dir_OS as Single, Position as Single, True_Dir as Single, Target_Dir as Single
Dim Icom3 as Byte, OCom3(1 to 40) as Byte
Dim CW as Boolean, MotorMoveFlag as Boolean, PotChangedFlag as Byte, AlreadyOnFlag as Boolean, OverCurrent_Count as Byte
Dim SW1 as Byte, Old_Meter as Byte
Dim ROT as Integer, TAR as Integer
Dim Target_Val as Single, Target_Diff as Single, Old_Target as Single, Target_Delta as Single, Target_Done as Byte, Read_Pos as Byte

Dim DirI as Integer
Const CWPin as Byte = 8
Const CCWPin as Byte = 9

Public Sub Main()
    CW = True
    'Set to move rotator clockwise
    MotorMoveFlag= False
    'Set rotator not to move
    AlReadyOnFlag = False
    'Flag motor not moving
    PotChangedFlag = 0
    'Flag pot not changed
    OverCurrent_Count = 0
    'Initialize over current counter
    Old_Meter = 0
    'Initialize old meter tic position to unused part of display
    Call PutPin(5, bxInputPullup)                    'Pin 5 is a pulled up
input
    Call PutPin(6, bxInputPullup)                    'Pin 6 is a pulled up
input
    Call PutPin(7, bxInputPullup)                    'Pin 7 is a pulled up
input
    Call PutPin(CWPin, bxOutputLow)                        'Motor off
clockwise
    Call PutPin(CCWPin, bxOutputLow)                 'Motor off
counterclockwise

'Set up COM3 port to drive LCD

    Call OpenQueue(OCom3, 40)                              'Open an
output queue for com3
    Call DefineCom3(0, 6, bx1000_1000)               'Com 3 as output on
pin 6 with 8 data bits, no parity
    Call OpenCom(3, 9600, ICom3, OCom3)              'Open com3
    Call Delay(0.5)
    Call PutQueueStr(OCom3, Chr(12))                 'Clear screen
    Call PutQueueStr(OCom3, Chr(2))                        'Set
brightness
    Call PutQueueStr(Ocom3, Chr(255))
    Call PutQueueStr(OCom3, Chr(3))                        'Set
contrast
    Call PutQueueStr(Ocom3, Chr(0))
    Call PutQueueStr(OCom3, Chr(23))                 'Set buzzer frequency
    Call PutQueueStr(Ocom3, Chr(150))
    Call PutQueueStr(Ocom3, Chr(16))                 'Position cursur to
element 3
    Call PutQueueStr(Ocom3, Chr(3))
    Call PutQueueStr(Ocom3, "TARGET ")               'Print target label
    Call PutQueueStr(Ocom3, " ACTUAL")               'Print rotator label

'Define Special Characters
    Dim N as Byte, M as Byte
'Special Character 0 is tic mark
    Call PutQueueStr(Ocom3, Chr(19))
    Call PutQueueStr(Ocom3, Chr(0))
    For N = 1 to 4
```

```
                Call PutQueueStr(Ocom3, Chr(0))
        Next
        For N = 1 to 3
                Call PutQueueStr(Ocom3, Chr(4))
        Next
        Call PutQueueStr(Ocom3, Chr(0))

    'Special Character 1 first meter character
        Call PutQueueStr(Ocom3, Chr(19))
        Call PutQueueStr(Ocom3, Chr(1))
        Call PutQueueStr(Ocom3, Chr(0))
        For N = 1 to 6
                Call PutQueueStr(Ocom3, Chr(16))
        Next
                Call PutQueueStr(Ocom3, Chr(0))

    'Special Character 2 second meter character
        Call PutQueueStr(Ocom3, Chr(19))
        Call PutQueueStr(Ocom3, Chr(2))
                Call PutQueueStr(Ocom3, Chr(0))
        For N = 1 to 6
                Call PutQueueStr(Ocom3, Chr(8))
        Next
                Call PutQueueStr(Ocom3, Chr(0))

    'Special Character 3 third meter character
        Call PutQueueStr(Ocom3, Chr(19))
        Call PutQueueStr(Ocom3, Chr(3))
                Call PutQueueStr(Ocom3, Chr(0))
        For N = 1 to 6
                Call PutQueueStr(Ocom3, Chr(4))
        Next
                Call PutQueueStr(Ocom3, Chr(0))

    'Special Character 4 fourth meter character
        Call PutQueueStr(Ocom3, Chr(19))
        Call PutQueueStr(Ocom3, Chr(4))
                Call PutQueueStr(Ocom3, Chr(0))
        For N = 1 to 6
                Call PutQueueStr(Ocom3, Chr(2))
        Next
                Call PutQueueStr(Ocom3, Chr(0))

    'Special Character 5 fifth meter character
        Call PutQueueStr(Ocom3, Chr(19))
        Call PutQueueStr(Ocom3, Chr(5))
                Call PutQueueStr(Ocom3, Chr(0))
        For N = 1 to 6
                Call PutQueueStr(Ocom3, Chr(1))
        Next
                Call PutQueueStr(Ocom3, Chr(0))

    'Read Center Direction Pins and determine offset
        SW1 = GetPin(5)
        If SW1 = 0 Then
                Dir_OS = 353.0
        'Center direction is North
                Call PutQueueStr(Ocom3, Chr(16))                                    'Position to last line
of display
                Call PutQueueStr(Ocom3, Chr(40))
                Call PutQueueStr(Ocom3, "E  S  W  N  E  S  W")                      'Display direction scale
        Else
                Dir_OS = 173.0
        'Center direction is South
                Call PutQueueStr(Ocom3, Chr(16))                                    'Position to next to
last line of display
```

```
                Call PutQueueStr(Ocom3, Chr(40))
                Call PutQueueStr(Ocom3, "W  N  E  S  W  N  E")                    'Display direction scale
        End If
    'Place scale tic marks
        For N = 0 to 5
                M = 41 + (3 * N)
                Call PutQueueStr(Ocom3, Chr(16))                                  'Position curser
                Call PutQueueStr(Ocom3, Chr(M))
                Call PutQueueStr(Ocom3, Chr(128))
                Call PutQueueStr(Ocom3, Chr(128))
        Next

    'This is the main loop that sees if a rotation needs to be done.  If the rotation sensor is moved, the
    'LED's increment or decrement to the desired heading, and the decimal points light.  The direction is determined by where the rotation pot is turned.
    'When the desired position is determined, the ROT button is pushed and the rotator tracks to the desired position and the decimal points are turned off.
    'The display now displays the actual position of the rotator in degrees.  The starting position of the rotator and the new desired position are used
    'to compute whether the rotator turns clockwise or counter clockwise to get to the desired position fastest without encountering the limits.
    'The LED's display the actual rotator position until the position pot is turned.  If the position pot is turned while the rotator is still turning from
    'the last command, the rotator stops.  If the ROT button is held in for 5 seconds, the LED's display the pot position in the rotator which goes
    'from 0 to 705 degrees.  If an over current condition is encountered, the rotator is stopped and the decimal points blink at a 1 second rate.
    'The power must be recycled to clear this condition.

        Do
                Call GetADC(14,Old_Target)
                'Read Target value
                Call Delay(0.1)
                Call DisplayDirection()
        'Display Target and Rotator directions
                Call GetDirection(True_Dir, Dir_OS,CW)
        'Get the position of rotator and compute the true direction
                Call CheckRotationSensor(True_Dir, Target_Dir, MotorMoveFlag)     'Check if a change in direction is desired
                Call MotorOn(True_Dir, Target_Dir, CW, MotorMoveFlag, AlreadyOnFlag)  'Move rotator to new position
                Call MotorCurrent(OverCurrent_Count)
        'Check for excess motor current
                Call MotorOff(True_Dir, Target_Dir, CW, AlReadyOnFlag)            'Stop motor if at target position
        Loop
    End Sub

    'GetDirection reads the rotator position pot, converts it to degrees and then to true degrees based on the center
    'position.  It also sets the limit flags.

    Sub GetDirection(True_Dir as Single, Dir_OS as Single, CW as Boolean)
        Dim CCW_SD as Single, CW_SD as Single, Position as Single
        CCW_SD = 83.0
        'Counter clockwise limit
        CW_SD = 622.0
        'Clockwise limit
        Call GetADC(13, Position)                                                 'Read pot votage as a single between 0.0 and 1.0
        Position = Position * 705.0                                               'Compute absolute position between 0 and 705 degrees
        'Set flag to rotate CW if Rotator at Counter Clockwise limit
        If Position <= CCW_SD Then
                CW = True
        End If
        'Set flag to rotate CCW if rotator is at Clockwise limit
        If Position >= CW_SD Then
                CW = False
```

```
        End If
        'Compute actual direction
        True_Dir = Position - Dir_OS
        If True_Dir < 0.0 Then
                True_Dir = True_Dir + 360.0
        End If
        If True_Dir >= 359.5 Then
                True_Dir = True_Dir - 360.0
        End If
    End Sub


    'DisplayDirection displays the direction of the rotator  and target in degrees on the LCD

    Sub DisplayDirection()
        Dim N as Byte, M as Byte, Pos as Integer
        TAR = CInt(Target_Dir)
        ROT = Cint(True_Dir)
        Call PutQueueStr(Ocom3, Chr(16))                                        'Position curser to
Target character
        Call PutQueueStr(OCom3, Chr(24))
        If PotChangedFlag = 0 Then                                                      'If Target
reached, blank Target value
                Call PutQueueStr(Ocom3, "---")
        Else
                'Otherwise, display Target value
                If TAR < 100 Then
                        Call PutQueueStr(OCom3, Chr(32))
                End If
                If TAR < 10 Then
                        Call PutQueueStr(OCom3, Chr(32))
                End If
                Call PutQueueStr(OCom3, Cstr(TAR))
        End If
        Call PutQueueStr(Ocom3, Chr(16))                                        'Position curser to
Rotator actual position
        Call PutQueueStr(OCom3, Chr(32))
        If ROT < 100 Then
        'Display rotator actual position
                Call PutQueueStr(OCom3, Chr(32))
        End If
        If ROT < 10 Then
                Call PutQueueStr(OCom3, Chr(32))
        End If
        Call PutQueueStr(OCom3, Cstr(ROT))
        Pos = CInt(Position)
        N = 0
        Do
                Pos = Pos - 30
                N = N + 1
        Loop Until Pos < 68
        N = N - 1
        M = N
        N = N + 60
        If N <> Old_Meter Then
                Call PutQueueStr(Ocom3, Chr(16))                                        'Position
to old meter tic position
                Call PutQueueStr(Ocom3, Chr(Old_Meter))
                Call PutQueueStr(Ocom3, " ")                                            'Blank old
tic
        End If
        Call PutQueueStr(Ocom3, Chr(16))                                                'Position
to new meter tic position
        Call PutQueueStr(Ocom3, Chr(N))
        Pos = 68 + (CInt(M) * 30)
        'Calculate fine tic position
```

```
        Select Case CInt(Position)
                Case Pos to Pos + 5
                        M = 1
                Case Pos + 6 to Pos + 11
                        M = 2
                Case Pos + 12 to Pos + 17
                        M = 3
                Case Pos + 18 to Pos + 23
                        M = 4
                Case Pos + 24 to Pos + 29
                        M = 5
        End Select
        M = M + 128
        Call PutQueueStr(Ocom3, Chr(M))
        'Output proper tic to display
        Old_Meter = N
                'Save old tic position for blanking if different next time
    End Sub

    'CheckRotationSensor checks to see if a rotation is desired and does it.  This routine looks at the value of
    'the pot and decides whether to increment or decrement and how fast to do it.

    Sub CheckRotationSensor(True_Dir as Single, Target_Dir as Single, MotorMoveFlag as Boolean)
        Dim Pos_Diff as Single
        Target_Delta = 0.01
        Call GetADC(14,Target_Val)
        'Read Target value
        Target_Diff = Target_Val - Old_Target
        Target_Diff = ABS(Target_Diff)
        If Target_Diff >= Target_Delta Then
                PotChangedFlag = 1
        'Flag that the Target pot has been changed
                If AlreadyOnFlag = True Then
                        Call RampDown(CW)
        'Turn motor off
                End If
                MotorMoveFlag= False
                Do
                        Call GetADC(14,Target_Val)
        'Read Target value
                        Target_Dir = (Target_Val - 0.5) * 360.0                                 'Convert to degrees
                        If Target_Dir < 0.0 Then
        'Adjust direction
                                Target_Dir = Target_Dir + 359.0
                        End If
                        Call DisplayDirection()                                                 'Display
Target Position
                        Target_Done = GetPin(7)
        'Check done
                        If Target_Done = 0 Then
                                MotorMoveFlag = True
                        End If
                Loop Until MotorMoveFlag = True
                Pos_Diff = Abs(Target_Dir - True_Dir)                                           'If Target Position
isn't 2 degrees from True Position, don't move motor
                If Pos_Diff <= 2.0 Then
                        MotorMoveFlag = False
                        PotChangedFlag = 0
        'Flag that the Target pot was not changed enough
                End If
                Call PutPin(16, bxOutputLow)                                                    'Turn off
decimal points
        End If
    End Sub
```

```
'MotorOn turns the rotator motor on after calculating the shortest way to go from where
'it is pointed to wher it needs to go.  If the shortest route violates the end travel of the
'rotator, then the longest route is implemented.  CW = True means go clockwise.  CW = False
'means go counterclockwise.

Sub MotorOn(True_Dir as Single, Target_Dir as Single, CW as Boolean, MotorMoveFlag as Boolean, AlreadyOnFlag as Boolean)
    Dim Pos_Diff as Single, Travel as Single
    If AlreadyOnFlag = False Then
            If MotorMoveFlag= True Then
    'Start the rotator motor if warrented
                If True_Dir >= 180.0 Then
                    If Target_Dir < 180.0 Then
                        Pos_Diff = True_Dir - Target_Dir
                        If Pos_Diff < 180.0 Then
                            CW = False
                            Travel = True_Dir - Target_Dir
                        Else
                            CW = True
                            Travel = 360.0 - True_Dir + Target_Dir
                        End If
                    Else
                        If True_Dir > Target_Dir Then
                            CW = False
                            Travel = True_Dir - Target_Dir
                        Else
                            CW = True
                            Travel = Target_Dir - True_Dir
                        End If
                    End If
                Else
                    If Target_Dir < 180.0 Then
                        If True_Dir > Target_Dir Then
                            CW = False
                            Travel = True_Dir - Target_Dir
                        Else
                            CW = True
                            Travel = Target_Dir - True_Dir
                        End If
                    Else
                        Pos_Diff = Target_Dir - True_Dir
                        If Pos_Diff < 180.0 Then
                            CW = True
                            Travel = Target_Dir - True_Dir
                        Else
                            CW = False
                            Travel = 360.0 - Target_Dir + True_Dir
                        End If
                    End If
                End If
                Call GetADC(13, Position)
    'Read pot votage as a single between 0.0 and 1.0
                Position = Position * 705.0
    'Compute absolute position between 0 and 705 degrees
                If Position >= 622.0 Then
    'Flag to go counterclockwise if at clockwise limit
                    CW = False
                End If
                If Position <= 83.0 Then
    'Flag to go clockwise if if at counterclockwise limit
                    CW = True
                End If
                If CW = True Then
                    If Position + Travel >= 622.0 Then
                        CW= False
                    End If
                Else
```

```
                                        If Position - Travel <= 83.0 Then
                                                CW = True
                                        End if
                                End if
                                Call RampUp(CW)
                        'Ramp up motor and leave running
                                AlReadyOnFlag = True
                'Flag motor as running
                        End If
                End If
                Call GetADC(13, Position)
                'Read pot votage as a single between 0.0 and 1.0
                Position = Position * 705.0
                'Compute absolute position between 0 and 705 degrees
                If CW = True Then
                        If Position >= 622.0 Then
                                Call RampDown(CW)
                        End If
                Else
                        If Position <= 83.0 Then
                                Call RampDown(CW)
                        End If
                End If
        End Sub

        'RampUp ramps the motor up to full speed over 100 msec and leaves it on

        Sub RampUp(CW as Boolean)
                Dim N as Byte, M as Byte, Q as Single, P as Single
                If CW = True Then
                        Call PutPin(CCWPin, bxOutputLow)                                        'Turn off
CounterClockwise Direction
        'Slowly ramp up motor speed ClockWise over 100 msec
                        For N = 1 to 9 Step 1
                                M = 10 - N
                                P = CSng(N) * 10.0e-3
                                Q = CSng(M) * 10.0e-3
                                Call PutPin(CWPin, bxOutputHigh)                                'Turn
motor on
                                Call Delay(P)
                                Call PutPin(CWPin, bxOutputLow)
                'Turn motor off
                                Call Delay(Q)
                                Call GetDirection(True_Dir, Dir_OS,CW)                  'Get the position of
rotator and compute the true direction
                                Call DisplayDirection()                                         'Display
true direction of rotator
                        Next
                        Call PutPin(CWPin, bxOutputHigh)                                        'Motor full
speed Clockwise
                Else
                        Call PutPin(CWPin, bxOutputLow)
                'Turn off Clockwise Direction
        'Slowly ramp up motor speed CounterClockWise over 100 msec
                        For N = 1 to 9 Step 1
                                M = 10 - N
                                P = CSng(N) * 10.0e-3
                                Q = CSng(M) * 10.0e-3
                                Call PutPin(CCWPin, bxOutputHigh)                               'Turn
motor on
                                Call Delay(P)
                                Call PutPin(CCWPin, bxOutputLow)                                'Turn
motor off
                                Call Delay(Q)
                                Call GetDirection(True_Dir, Dir_OS,CW)                  'Get the position of
rotator and compute the true direction
```

```
                Call DisplayDirection()                                         'Display
true direction of rotator
            Next
            Call PutPin(CCWPin, bxOutputHigh)                                   'Motor full
speed Counterclockwise
        End If
    End Sub

    'RampDown ramps the motor down to zero speed over 100 msec and leaves it off

    Sub RampDown(CW as Boolean)
        Dim N as Byte, M as Byte, Q as Single, P as Single
        If CW = True Then
            Call PutPin(CCWPin, bxOutputLow)                                    'Turn off
CounterClockwise Direction
    'Slowly ramp down motor speed ClockWise over 100 msec
            For N = 9 to 1 Step -1
                M = 10 - N
                P = CSng(N) * 10.0e-3
                Q = CSng(M) * 10.0e-3
                Call PutPin(CWPin, bxOutputHigh)                                'Turn
motor on
                Call Delay(P)
                Call PutPin(CWPin, bxOutputLow)
        'Turn motor off
                Call Delay(Q)
                Call GetDirection(True_Dir, Dir_OS,CW)                          'Get the position of
rotator and compute the true direction
                Call DisplayDirection()                                         'Display
true direction of rotator
            Next
        Else
            Call PutPin(CWPin, bxOutputLow)
        'Turn off Clockwise Direction
    'Slowly ramp down motor speed CounterClockWise over 100 msec
            For N = 9 to 1 Step -1
                M = 10 - N
                P = CSng(N) * 10.0e-3
                Q = CSng(M) * 10.0e-3
                Call PutPin(CCWPin, bxOutputHigh)                               'Turn
motor on
                Call Delay(P)
                Call PutPin(CCWPin, bxOutputLow)                                'Turn
motor off
                Call Delay(Q)
                Call GetDirection(True_Dir, Dir_OS,CW)                          'Get the position of
rotator and compute the true direction
                Call DisplayDirection()                                         'Display
true direction of rotator
            Next
        End If
        Call PutPin(CWPin, bxOutputLow)                                         'Motor stopped
        Call PutPin(CCWPin, bxOutputLow)                                    'Motor stopped
        AlreadyOnFlag = False
        'Flag motor stopped
    End Sub

    'MotorOff stops the motor if the rotator has reached the target position or a new target is selected

    Sub MotorOff(True_Dir as Single, Target_Dir as Single, CW as Boolean, AlReadyOnFlag as Boolean)
        Dim Pos_Diff as Single
        If AlReadyOnFlag = True Then
            Pos_Diff = Abs(Target_Dir - True_Dir)
            If Pos_Diff <= 5.0 Then
        'Close enough, stop motor
```

```
                    Call RampDown(CW)
        'Stop slowly
                    AlreadyOnFlag = False
        'Flag motor not already on
                    MotorMoveFlag = False
        'Flag do not move motor
                    PotChangedFlag = 0
        'Flag that Target is reached, so blank Target reading
                End If
        End If
    End Sub

    'MotorCurrent monitors the motor current.  If the current exceeds 5 amps for 500 msec, this routine shuts off
    'the motor, flashes the decimal points, and needs a power on/off to reset.

    Sub MotorCurrent(OverCurrent_Count as Byte)
        Dim Motor_Current as Single
        Call GetADC(15, Motor_Current)
        If Motor_Current >= 0.1 Then
                OverCurrent_Count = OverCurrent_Count +1
        Else
                OverCurrent_Count = 0
        End If
        If OverCurrent_Count > 3 Then                                                   'turn off if
over current for ~3 seconds
                Call PutPin(CWPin, bxOutputLow)
        'Motor off clockwise
                Call PutPin(CCWPin, bxOutputLow)                                        'Motor off
counterclockwise
                Do
                        Call PutQueueStr(Ocom3, Chr(16))                                'Position
cursor
                        Call PutQueueStr(Ocom3, Chr(38))
                        Call PutQueueStr(Ocom3, "OC")                                   'Display
Over Current sign
                        Call PutQueueStr(Ocom3, Chr(7))                                 'Beep
                        Call Delay(1.0)
        'Delay 1 second
                        Call PutQueueStr(Ocom3, Chr(16))                                'Position
cursor
                        Call PutQueueStr(Ocom3, Chr(38))
                        Call PutQueueStr(Ocom3, "  ")                                 'Blank
Over Current sign
                        Call PutQueueStr(Ocom3, Chr(7))                                 'Beep
                        Call Delay(1.0)
        'Delay 1 second
                Loop
        End If
    End Sub
```